# Multi-proxy Multi-signature without Pairing from Certificateless Cryptography

Rena Ehmet[1,2], Lunzhi Deng[3,4,5], Yingying Zhang[1], Jiwen Zeng[1]
*(Corresponding author: Lunzhi Deng)*

School of Mathematical Sciences, Xiamen University, China[1]
School of Mathematical Sciences, Xinjiang Normal University, China[2]
School of Mathematical Sciences, Guizhou Normal University, China[3]
School of Computer Science, Guizhou University, China[4]
Guizhou University, Guizhou Provincial Key Laboratory of Public Big Data, China[5]
(Email: denglunzhi@163.com)
*(Received Oct. 11, 2016; revised and accepted Feb. 20, 2017)*

## Abstract

In a multi-proxy multi-signature scheme, there is a group of original signers who delegate their signing rights to another group of persons called proxy group. Most of the known cryptography schemes used bilinear pairings, the computation cost of the which is much higher than that of the exponentiation in a RSA group. In this paper, we propose a certificateless multi-proxy multi-signature scheme based on the classic RSA and discrete logarithm (DL) problem. Our scheme is also constructed without using pairing which reduces the running time significantly, and it is secure against chosen message attack in random oracle model and more applicable for practical applications.

*Keywords: Certificateless Cryptography; Multi-proxy; Multi-signature; RSA*

## 1 Introduction

In traditional public key cryptography, the users first need to obtain the authenticated public keys from a certificate authority, if they want to communicate a message. In that system, the certificate management, storage space and large overhead to transfer certificates lead to increase the associated communication cost.

**ID-Based Cryptography.** To solve the certificate management problem in the traditional public key cryptography, Shamir [16] introduced the ID-based cryptography in 1984, which removed the need of certificate for public key and thus reduced the associated communication cost. In ID-based cryptography, the users' public and private keys are generated from their identities such as email addresses, IP addresses, etc. There is a very important problem in ID-based cryptosystem that user's private key is generated by a key generation center (KGC). It means that KGC knows user's private key. So ID-based public key cryptography has to face with key escrow problem.

**Certificateless Cryptography.** In 2003, Al-Riyami *et al.* [1] proposed the concept of certificateless public key cryptosystem (CLPKC). In CLPKC, a user's private key is comprised of partial private key generated by KGC and a secret value chosen by the user separately. The certificateless public key cryptography has attracted much attention since it solves the certificate management problem in the traditional public cryptography and the key escrow problem in the ID-based cryptography.

**Proxy Signatures.** In 1996, Mambo *et al.* proposed the first proxy signature scheme [13], which allows an entity, called original signer, to delegate his/her signing right to other entity, called proxy signer. The multi-proxy signature scheme allows a group of proxy signers generate signatures, on behalf of one original signer — a company or an organization, who delegates his/her signing right to the proxy group [9, 11, 19]. Multi-proxy multi-signature (MPMS) is a new kind of proxy signature, firstly proposed by Tzeng *et al.* [20] in 2004, in which a group of original signers can authorize a group of proxy signers under the agreement of all original and proxy signers, so that a signature could be generated by the cooperation of all proxy signers. It solves many real life problems. For example in a company, there are some conflict between the employees and the employers. All employees want to depute a lawyer group as their proxy agents.

**Cryptography from RSA.** RSA is one of the first practical asymmetric public-key cryptosystems and widely used for secure data transmission. In RSA cryptosystem, its asymmetry is based on the practical difficulty of factoring the product of two large prime numbers. Being as a classified difficult problem, RSA is widely used in many aspects of cryptography. Shamir [16] constructed the first ID-based signature scheme from RSA in 1985.

Herranz [7] proposed an ID-based ring signature scheme whose security is based on the hardness of RSA problem.

Using bilinear pairings, people proposed many new proxy signature scheme [2, 10, 12, 14, 15, 21, 22, 23]. All the above schemes are very practical, but they used bilinear pairings and the pairing is regarded as the most expensive cryptography primitive. In 2011, He *et al.* [5] proposed an ID-based proxy signature scheme without bilinear pairing. In 2013, He *et al.* [6] put forward a certificateless proxy signature scheme without bilinear pairing. Kim *et al.* [8] constructed a provably secure ID-based proxy signature scheme based on the lattice problems. In 2014, Deng *et al.* [4] constructed a certificateless proxy signature based on RSA and discrete logarithm problem. In 2015, Tiwari and Padhye [17] proposed a provable secure multi-proxy signature scheme without bilinear map. In 2017, Deng *et al.* [3] put forward an ID-based proxy signature from RSA without bilinear pairing. The computation cost of the pairing is much higher than that of the exponentiation in a RSA group. Therefore, certificateless schemes based on RSA primitive would still be appealing.

**Our Contribution.** By using the idea from [4], we propose a new certificateless multi-proxy multi-signature (CLMPMS) scheme. Security of the scheme is based on the famous RSA problem and DL (discrete-logarithm) problem. And our scheme is efficient in reducing the running time significantly because of its pairing-freeness. In addition, we analyze the security of our scheme against both of the super Type I and the super Type II adversaries. To the best of authors' knowledge, our scheme is the first certificateless multi-proxy multi-signature based on RSA and DL problem.

**Roadmap.** The organization of the paper is sketched as follows: The Section 2 gives some preliminaries and the formal model. We present our proposed scheme in Section 3. The security analysis and performance comparisons will be given in Sections 4 and 5 separately. Finally, we give some conclusions in Section 6.

# 2 Preliminaries

## 2.1 Elliptic Curve Group

Let $E/F_p$ denote an elliptic curve $E$ over a prime finite field $F_p$, defined by an equation

$$y^2 = x^3 + ax + b(\mathrm{mod}p),\ a, b \in F_p \text{ and } 4a^3 + 27b^2 \neq 0(\mathrm{mod}p),$$

The points on $E/F_p$ together with an extra point $O$ called the point at infinity form a group

$$\mathcal{G} = \{(x, y) : x, y \in F_p, E(x, y) = 0\} \cup \{O\}.$$

## 2.2 Notations

- $N$: A large composite number, the product of two prime numbers $p, q$.
- $\mathbb{G}$: A cyclic subgroup of $\mathcal{G}$ with prime order $b$ and $gcd(b, \varphi(N)) = 1$.
- $P$: A generator of group $\mathbb{G}$.
- $D_i$: The partial private key of user $ID_i$ generated by KGC.
- $t_i$: The secret value chosen by user $ID_i$.
- $P_i$: The public key of user $ID_i$.
- RSAP: Given a tuple $(N, b, \mathcal{Y})$ to find $\mathcal{X} \in \mathbb{Z}_N^*$ such that $\mathcal{X}^b = \mathcal{Y} \bmod N$.
- DLP: Given a tuple $(P, xP)$ in $\mathbb{G}$ to compute $x \in Z_b^*$.

## 2.3 System Model

The proposed model involves four parties: a set of $n$ original signers $\mathcal{N} = \{ID_{o1}, ID_{o2}, \cdots, ID_{on}\}$, a set of $l$ proxy signers $\mathcal{L} = \{ID_{p1}, ID_{p2}, \cdots, ID_{pl}\}$, a verifier $\mathcal{V}$, and a clerk $\mathcal{B}$. Use of clerk reduces the communication cost.

**Definition 1.** *A multi-proxy multi-signature scheme is specified by the following polynomial time algorithms.*

**Setup.** Given a security parameter $k$, this algorithm outputs the system parameters *params*, and keeps msk as system's master secret key.

**Partial private key extract.** Given an identity $ID_i \in \{0, 1\}^*$, the master secret key $msk$, and parameters *params*, the key generation center (KGC) generates the partial private key $D_i$ for the identity $ID_i$.

**Set secret value.** The user with identity $ID_i$ chooses a random number as his secret value.

**User's public key generation.** The user with identity $ID_i$ computes his public key.

**Proxy certificate generation.** This algorithm takes the warrant $m_\omega$ to be signed and generates the proxy certificate with the cooperation of all original signers and proxy signers.

**Multi-proxy sign.** This algorithm takes the certificate and a message $M \in \{0, 1\}^*$ as input, and outputs a multi-proxy multi-signature signed by the proxy group $\mathcal{L}$ on behalf of the original group $\mathcal{N}$.

**Verify.** This algorithm takes the identities of all original signers, the identities of all proxy signers, and a proxy signature as input, returns *True* if it is a valid signature on $M$ signed by the proxy group $\mathcal{L}$, on behalf of the original group $\mathcal{N}$. Otherwise, returns $False$.

## 2.4 Security Model

For a certificateless multi-proxy multi-signature scheme, there are two kinds of adversaries. The adversary $\mathcal{A}_1$ is not able to access the master key, but he could replace users' public keys at his will. The adversary $\mathcal{A}_2$ can access the master key, but he is not able to replace users' public keys. The security of CLMPMS schemes are formally defined through two games played between a challenger $\mathcal{C}$ and an adversary $\mathcal{A} \in \{\mathcal{A}_1,\ \mathcal{A}_2\}$.

**Definition 2.** *A CLMPMS scheme is unforgeable if no polynomially bounded adversary has a non-negligible advantage in the following two games against Type I and Type II adversaries.*

**Game I:** It performs between the challenger $\mathcal{C}$ and a Type I adversary $\mathcal{A}_1$ for the CLMPMS scheme.

**Initialization:** $\mathcal{C}$ runs the setup algorithm, takes a security parameter $k$ as input to obtain a master key $msk$ and the system parameters $params$. $\mathcal{C}$ then sends $params$ to the adversary $\mathcal{A}_1$ and keeps $msk$ secret. The point is that adversary $\mathcal{A}_1$ doesn't know $msk$.

**Queries:** $\mathcal{A}_1$ can get access to query the following oracles polynomially bounded number of times which are controlled by $\mathcal{C}$. Each query may depend on the answers to the previous query.

- User-Public-Key-Oracle: This oracle takes a user's identity $ID_i$ as input. If $ID_i$'s public key has already been queried, nothing is to be carried out. Otherwise, it generates the secret value $t_i$ and the public key $P_i$. Then it returns $P_i$ and adds $(ID_i, D_i, t_i, P_i)$ to the list $L_U$.
- Partial-Private-Key-Oracle: On inputting an identity $ID_i$, the oracle browses the list $L_U$ and returns the partial private key $D_i$ as answer. Otherwise, returns 0.
- Public-Key-Replacement-Oracle: Taking an identity $ID_i$ and a new public key $P_i'$ as input, the oracle replaces the public key of the given identity $ID_i$ with new one and updates the corresponding information in the list $L_U$.
- Secret-Value-Oracle: On inputting a created identity $ID_i$, the oracle browses the list $L_U$ and returns the secret value $t_i$ as answer. Otherwise returns 0. Note that $t_i$ is the secret value associated with the original public key $P_i$. $\mathcal{A}_1$ can't query the secret value for $ID_i$ whose public key has been replaced.
- Proxy-Certificate-Generation-Oracle: When $\mathcal{A}_1$ submits all signers' identities/public keys $(ID_i, P_i),\ ID_i \in \mathcal{N} \cup \mathcal{L}$ and a warrant $m_\omega$ to the challenger, $\mathcal{C}$ responds by running the proxy certificate generation algorithm on the warrant $m_\omega$ and the signers' full private keys $(t_i, D_i),\ ID_i \in \mathcal{N} \cup \mathcal{L}$.
- Proxy-Sign-Oracle: When $\mathcal{A}_1$ submits certificate $\pi$ and a message $M$ to the challenger, $\mathcal{C}$ responds by running the proxy sign algorithm on $\pi$, $M$ and the proxy signers' full private keys $(t_i, D_i),\ ID_i \in \mathcal{L}$.

**Forge:** $\mathcal{A}_1$ outputs a tuple $(\pi^*, \mathcal{N} \cup \mathcal{L}, \bigcup_{ID_i \in \mathcal{N} \cup \mathcal{L}} P_i)$ or $(M^*, m_\omega^*, \sigma^*, \mathcal{N} \cup \mathcal{L}, \bigcup_{ID_i \in \mathcal{N} \cup \mathcal{L}} P_i)$. We say $\mathcal{A}_1$ wins the game, if one of the following cases is satisfied:

**Case 1:** $\mathcal{A}_1$ outputs a tuple $(\pi^*, \mathcal{N} \cup \mathcal{L}, \bigcup_{ID_i \in \mathcal{N} \cup \mathcal{L}} P_i)$ satisfying:

1) $\pi^*$ is a valid certificate.
2) $\pi^*$ is not generated from the certificate generation query.
3) There is at least one user $ID \in \mathcal{N} \cup \mathcal{L}$ whose partial private key is not queried by $\mathcal{A}_1$.

**Case 2:** $\mathcal{A}_1$ outputs a tuple $(M^*, m_\omega^*, \sigma^*, \mathcal{N} \cup \mathcal{L}, \bigcup_{ID_i \in \mathcal{N} \cup \mathcal{L}} P_i)$ satisfying:

1) $\sigma^*$ is a valid proxy signature.
2) $\sigma^*$ is not generated from the proxy signature query.
3) $(m_\omega^*, \mathcal{N} \cup \mathcal{L}, \bigcup_{ID_i \in \mathcal{N} \cup \mathcal{L}} P_i)$ didn't appear in the certificate generation query.
4) There is at least one user $ID \in \mathcal{N}$ whose partial private key is not queried by $\mathcal{A}_1$.

**Case 3:** $\mathcal{A}_1$ outputs a tuple $(M^*, m_\omega^*, \sigma^*, \mathcal{N} \cup \mathcal{L}, \bigcup_{ID_i \in \mathcal{N} \cup \mathcal{L}} P_i)$ satisfying:

1) $\sigma^*$ is a valid proxy signature.
2) $\sigma^*$ is not generated from the proxy signature query.
3) There is at least one user $ID \in \mathcal{L}$ whose partial private key is not queried by $\mathcal{A}_1$.

The advantage of $\mathcal{A}_1$ is defined as

$$Adv_{\mathcal{A}_1}^{UNF-CLMPMS} = Pr[\mathcal{A}_1\ wins].$$

**Game II:** It performs between the challenger $\mathcal{C}$ and a Type II adversary $\mathcal{A}_2$ for the CLMPMS scheme.

**Initialization.** $\mathcal{C}$ runs the setup algorithm, takes a security parameter $k$ as input to obtain a master key $msk$ and the system parameters $params$. $\mathcal{C}$ then sends $msk$, $params$ to the adversary $\mathcal{A}_2$. It means that in Game II adversary $\mathcal{A}_2$ knows $msk$, he/she just can't replace the public key.

**Queries.** $\mathcal{A}_2$ may adaptively make a polynomially bounded number of queries as in Game I.

**Forge.** $\mathcal{A}_2$ outputs a tuple $(\pi^*, \mathcal{N} \cup \mathcal{L}, \bigcup_{ID_i \in \mathcal{N} \cup \mathcal{L}} P_i)$ or $(M^*, m_\omega^*, \sigma^*, \mathcal{N} \cup \mathcal{L}, \bigcup_{ID_i \in \mathcal{N} \cup \mathcal{L}} P_i)$. We say $\mathcal{A}_2$ wins the game, if one of the following cases is satisfied:

**Case 1:** $\mathcal{A}_2$ outputs a tuple $(\pi^*, \mathcal{N} \cup \mathcal{L}, \bigcup_{ID_i \in \mathcal{N} \cup \mathcal{L}} P_i)$ satisfying:

1) $\pi^*$ is a valid certificate.

2) $\pi^*$ is not generated from the certificate generation query.
3) There is at least one user $ID \in \mathcal{N} \cup \mathcal{L}$ whose secret value is not queried and whose public key is not placed by $\mathcal{A}_2$.
4) $\mathcal{A}_2$ can't query the secret value for any identity if the corresponding public key has already been replaced.

**Case 2:** If $\mathcal{A}_2$ outputs a tuple $(M^*, m_\omega^*, \sigma^*, \mathcal{N} \cup \mathcal{L}, \bigcup_{ID_i \in \mathcal{N} \cup \mathcal{L}} P_i)$ satisfying:
1) $\sigma^*$ is a valid proxy signature.
2) $\sigma^*$ is not generated from the proxy signature query.
3) $(m_\omega^*, \mathcal{N} \cup \mathcal{L}, \bigcup_{ID_i \in \mathcal{N} \cup \mathcal{L}} P_i)$ didn't appear in the certificate generation query.
4) There is at least one user $ID \in \mathcal{N}$ whose secret value is not queried and whose public key is not placed by $\mathcal{A}_2$.
5) $\mathcal{A}_2$ can't query the secret value for any identity if the corresponding public key has been replaced.

**Case 3:** If $\mathcal{A}_2$ outputs a tuple $(M^*, m_\omega^*, \sigma^*, \mathcal{N} \cup \mathcal{L}, \bigcup_{ID_i \in \mathcal{N} \cup \mathcal{L}} P_i)$ satisfying:
1) $\sigma^*$ is a valid proxy signature.
2) $\sigma^*$ is not generated from the proxy signature query.
3) There is at least one user $ID \in \mathcal{L}$ whose secret is not queried and whose public key is not placed by $\mathcal{A}_2$.
4) $\mathcal{A}_2$ can't query the secret value for any identity if the corresponding public key has already been replaced.

The advantage of $\mathcal{A}_2$ is defined as

$$Adv_{\mathcal{A}_2}^{UNF-CLMPMS} = Pr[\mathcal{A}_2 \ wins].$$

## 3 Our Scheme

In this section we will propose a certificateless multi-proxy multi-signature scheme based on RSA problem and DL problem, with the clerk architecture and without pairings. The scheme involves a set of $n$ original signers $\mathcal{N} = \{ID_{o1}, ID_{o2}, \cdots, ID_{on}\}$, a set of $l$ proxy signers $\mathcal{L} = \{ID_{p1}, ID_{p2}, \cdots, ID_{pl}\}$, a verifier $\mathcal{V}$ and a clerk $\mathcal{B}$. A cooperative clerk reduces the communication cost. Our scheme is described as follows:

**Setup.** Given a security parameter $k$, KGC generates two random $k$-bit prime numbers $p$ and $q$, then it computes $N = pq$. For some fixed parameter $m$ (for example $m = 200$), KGC randomly chooses a prime number $b$ satisfying $2^m < b < 2^{m+1}$ and $\gcd(b, \varphi(N)) = 1$. Then it chooses group $\mathbb{G}$ of prime order $b$, generator $P$ of $\mathbb{G}$, and computes $a = b^{-1} \bmod \varphi(N)$. Furthermore, KGC chooses five hash functions as follows: $H_0 = \{0,1\}^* \to Z_N^*$, $H_i : \{0,1\}^* \to Z_b^*$ $(i = 1,2,3,4)$. Finally KGC outputs the set of public parameters $params = \{N, b, \mathbb{G}, P, H_0, H_1, H_2, H_3, H_4\}$, and the master secret key $msk = (p, q, a)$.

**Partial private key extract.** For an identity $ID_i \in \{0,1\}^*$, KGC computes $Q_i = H_0(ID_i)$, $D_i = Q_i^a$, then sends $D_i$ to the user $ID_i$ via secure channel.

**Secret value set.** The user with identity $ID_i \in \{0,1\}^*$ randomly chooses $t_i \in Z_b^*$.

**Public key generation.** The user with identity $ID_i \in \{0,1\}^*$ computes his public key $P_i = t_i P$.

**Proxy certificate generation.** $m_\omega$ is the warrant consisting of the identities of $n$ original signers $ID_{oi}$ $(i = 1,2,\cdots,n)$, $l$ proxy signers $ID_{pj}$ $(j = 1,2,\cdots,l)$, the certificate during and so on. On inputting the warrant $m_\omega$, all signers $ID_{oi}$ $(i = 1,2,\cdots,n)$ and $ID_{pj}$ $(j = 1,2,\cdots,l)$ perform the following steps:

- Each $ID_{oi}$ randomly selects $c_{oi} \in Z_b^*$, $A_{oi} \in Z_N^*$. Computes $S_{oi} = c_{oi}P$, $T_{oi} = A_{oi}^b \bmod N$. Broadcasts $(S_{oi}, T_{oi})$ to the other $n-1$ original signers, $l$ proxy signers and clerk $\mathcal{B}$.
- Each $ID_{pj}$ randomly selects $c_{pj} \in Z_b^*$, $A_{pj} \in Z_N^*$. Computes $S_{pj} = c_{pj}P$, $T_{p_j} = A_{p_j}^b \bmod N$. Broadcasts $(S_{pj}, T_{pj})$ to the other $n$ original signers, $l-1$ proxy signers and clerk $\mathcal{B}$.
- Clerk $\mathcal{B}$ and all signers compute $S = \sum_{i=1}^n S_{oi} + \sum_{j=1}^l S_{pj}$, $T = \prod_{i=1}^n T_{oi} \cdot \prod_{j=1}^l T_{pj}$.
- Each $ID_{oi}$ computes

$$\begin{aligned} k_{oi} &= H_1(m_\omega, ID_{oi}, P_{oi}, S, T), \\ h_{oi} &= H_2(m_\omega, ID_{oi}, P_{oi}, S, T). \end{aligned}$$

  Computes $r_{oi} = c_{oi} + t_{oi}k_{oi}$, $R_{oi} = A_{oi}D_{oi}^{h_{oi}} \bmod N$. Broadcasts $(r_{oi}, R_{oi})$ to clerk $\mathcal{B}$.
- Each $ID_{pj}$ computes

$$\begin{aligned} k_{p_j} &= H_1(m_\omega, ID_{pj}, P_{pj}, S, T), \\ h_{pj} &= H_2(m_\omega, ID_{pj}, P_{pj}, S, T). \end{aligned}$$

  Computes $r_{pj} = c_{pj} + t_{pj}k_{pj}$, $R_{pj} = A_{pj}D_{pj}^{h_{pj}} \bmod N$. Broadcasts $(r_{pj}, R_{pj})$ to clerk $\mathcal{B}$.
- Clerk $\mathcal{B}$ does as follows:
  1) Verifies the correctness of $r_{oi}, R_{oi}$ by checking the equations: $r_{oi}P = S_{oi} + k_{oi}P_{oi}$, $R_{oi}^b = T_{oi}Q_{oi}^{h_{oi}} \bmod N$ for $ID_{oi} \in \mathcal{N}$.
  2) Verifies the correctness of $r_{pj}, R_{pj}$ by checking the equations: $r_{pj}P = S_{pj} + k_{pj}P_{pj}$, $R_{pj}^b = T_{pj}Q_{pj}^{h_{pj}} \bmod N$ for $ID_{pj} \in \mathcal{L}$.
  3) If all equalities hold, $\mathcal{B}$ computes

$$r = \sum_{i=1}^n r_{oi} + \sum_{j=1}^l r_{pj}, \ R = \prod_{i=1}^n R_{oi} \cdot \prod_{j=1}^l R_{pj}.$$

4) Sends $\pi = (m_\omega, r, R, S, T)$ to $n$ original signers and $l$ proxy signers.

**Multi-proxy multi-sign.** To sign a message $M$ on behalf of the $n$ original signers, the $l$ proxy signers perform the following steps:

- Each proxy signer $ID_{pj}$ selects $a_j \in Z_b^*$, $B_j \in Z_N^*$. Computes $X_j = a_j P$, $Y_j = B_j^b \bmod N$. Broadcasts $(X_j, Y_j)$ to the other $l-1$ proxy signers.
- Each proxy signer $ID_{pj}$ computes

$$\begin{aligned} X &= \sum_{j=1}^{l} X_j, \\ Y &= \prod_{j=1}^{l} Y_j. \\ \alpha_j &= H_3(M, m_\omega, ID_{pj}, P_{pj}, S, T, X, Y), \\ \beta_j &= H_4(M, m_\omega, ID_{pj}, P_{pj}, S, T, X, Y), \\ u_j &= r + a_j + t_{pj}\alpha_j, \\ U_j &= RB_j D_{pj}^{\beta_j}. \end{aligned}$$

Sends $(M, m_\omega, u_j, U_j, X_j, Y_j, S, T)$ to clerk $\mathcal{B}$.

- The clerk $\mathcal{B}$ checks whether all the proxy signers' partial signatures are correct.
  1) Computes $X = \sum_{j=1}^{l} X_j$, $Y = \prod_{j=1}^{l} Y_j$.
  2) Computes

$$\begin{aligned} k_{oi} &= H_1(m_\omega, ID_{oi}, P_{oi}, S, T), \\ h_{oi} &= H_2(m_\omega, ID_{oi}, P_{oi}, S, T) \\ &\qquad \text{for } i = 1, 2, \cdots, n. \\ k_{pj} &= H_1(m_\omega, ID_{pj}, P_{pj}, S, T), \\ h_{pj} &= H_2(m_\omega, ID_{pj}, P_{pj}, S, T) \\ &\qquad \text{for } j = 1, 2, \cdots, l. \\ \alpha_j &= H_3(M, m_\omega, ID_{pj}, P_{pj}, S, T, X, Y), \\ \beta_j &= H_4(M, m_\omega, ID_{pj}, P_{pj}, S, T, X, Y) \\ &\qquad \text{for } j = 1, 2, \cdots, l. \end{aligned}$$

  3) Computes

$$\begin{aligned} V &= \sum_{i=1}^{n} k_{oi} P_{oi} + \sum_{j=1}^{l} k_{pj} P_{pj}, \\ W &= \prod_{i=1}^{n} Q_{oi}^{h_{oi}} \cdot \prod_{j=1}^{l} Q_{pj}^{h_{pj}}. \end{aligned}$$

  4) Checks whether

$$\begin{aligned} u_j P &= S + V + X_j + \alpha_j P_{pj}, \\ U_j^b &= T \cdot W \cdot Y_j \cdot Q_{pj}^{\beta_j} \text{ for each } ID_j \in \mathcal{L}. \end{aligned}$$

If all equations hold, the clerk computes $u = \sum_{j=1}^{l} u_j$ and $U = \prod_{j=1}^{l} U_j$.

- Outputs multi-proxy multi-signature $\sigma = (M, m_\omega, u, U, S, T, X, Y)$.

**Multi-proxy multi-signature verify.** To verify the validity of the signature $\sigma = (M, m_\omega, u, U, S, T, X, Y)$ on message $M$, the verifier does as follows:

1) Checks whether the message $M$ conforms to the warrant $m_\omega$. If not, stops. Otherwise, continues.
2) Checks whether the $l$ proxy signers are authorized by the original group $\mathcal{N}$ in the warrant $m_\omega$. If not, stops. Otherwise, continues.
3) Computes

$$\begin{aligned} k_{oi} &= H_1(m_\omega, ID_{oi}, P_{oi}, S, T), \\ h_{oi} &= H_2(m_\omega, ID_{oi}, P_{oi}, S, T) \\ &\qquad \text{for } i = 1, 2, \cdots, n. \\ k_{pj} &= H_1(m_\omega, ID_{pj}, P_{pj}, S, T), \\ h_{pj} &= H_2(m_\omega, ID_{pj}, P_{pj}, S, T) \\ &\qquad \text{for } j = 1, 2, \cdots, l. \\ \alpha_j &= H_3(M, m_\omega, ID_{pj}, P_{pj}, S, T, X, Y), \\ \beta_j &= H_4(M, m_\omega, ID_{pj}, P_{pj}, S, T, X, Y) \\ &\qquad \text{for } j = 1, 2, \cdots, l. \end{aligned}$$

4) Computes

$$\begin{aligned} V &= \sum_{i=1}^{n} k_{oi} P_{oi} + \sum_{j=1}^{l} k_{pj} P_{pj}, \\ W &= \prod_{i=1}^{n} Q_{oi}^{h_{oi}} \cdot \prod_{j=1}^{l} Q_{pj}^{h_{pj}}. \end{aligned}$$

5) Checks whether the equations below hold. If both hold, accepts. Otherwise, rejects.

$$\begin{aligned} uP &= l(S + V) + X + \sum_{j=1}^{l} (\alpha_j P_{pj}), \\ U^b &= (TW)^l \cdot Y \cdot \prod_{j=1}^{l} Q_{pj}^{\beta_j}. \end{aligned}$$

**Correctness:**

$$\begin{aligned} uP &= \sum_{j=1}^{l} u_j P = \sum_{j=1}^{l} \left(r + a_j + t_{pj}\alpha_j\right) P \\ &= lrP + \sum_{j=1}^{l} (a_j P + \alpha_j t_{pj} P) \\ &= lrP + \sum_{j=1}^{l} (X_j + \alpha_j P_{pj}) \\ &= l(\sum_{i=1}^{n} r_{oi} + \sum_{j=1}^{l} r_{pj})P + \sum_{j=1}^{l} (X_j + \alpha_j P_{pj}) \\ &= \sum_{i=1}^{n} l r_{oi} P + \sum_{j=1}^{l} (l r_{pj} P + X_j + \alpha_j P_{pj}) \end{aligned}$$

$$
\begin{aligned}
&= \sum_{i=1}^{n} l(c_{oi} + t_{oi}k_{oi})P \\
&\quad + \sum_{j=1}^{l}(l(c_{pj} + t_{pj}k_{pj})P + X_j + \alpha_j P_{pj}) \\
&= \sum_{i=1}^{n}(lS_{oi} + lk_{oi}P_{oi}) \\
&\quad + \sum_{j=1}^{l}(lS_{pj} + lk_{pj}P_{pj} + X_j + \alpha_j P_{pj}) \\
&= lS + X + \sum_{i=1}^{n} lk_{oi}P_{oi} + \sum_{j=1}^{l}(lk_{pj} + \alpha_j)P_{pj} \\
&= l(S+V) + X + \sum_{j=1}^{l}(\alpha_j P_{pj}).
\end{aligned}
$$

$$
\begin{aligned}
U^b &= \big(\prod_{j=1}^{l} U_j\big)^b = \prod_{j=1}^{l}\big(RB_j D_{pj}^{\beta_j}\big)^b = R^{lb}\cdot\prod_{j=1}^{l} Y_j Q_{pj}^{\beta_j} \\
&= (\prod_{i=1}^{n} R_{oi})^{lb}\cdot(\prod_{j=1}^{l} R_{pj})^{lb}\cdot\prod_{j=1}^{l} Y_j Q_{pj}^{\beta_j} \\
&= \prod_{i=1}^{n}(A_{oi}D_{oi}^{h_{oi}})^{lb}\cdot\prod_{j=1}^{l}(A_{pj}D_{pj}^{h_{pj}})^{lb} Y_j Q_{pj}^{\beta_j} \\
&= \prod_{i=1}^{n} T_{oi}^{l} Q_{oi}^{lh_{oi}}\cdot\prod_{j=1}^{l} T_{pj}^{l} Q_{P_j}^{lh_{pj}+\beta_j} Y_j \\
&= T^l\cdot Y\cdot\prod_{i=1}^{n} Q_{oi}^{lh_{oi}}\cdot\prod_{j=1}^{l} Q_{pj}^{lh_{pj}+\beta_j} \\
&= (TW)^l\cdot Y\cdot\prod_{j=1}^{l} Q_{pj}^{\beta_j}.
\end{aligned}
$$

# 4 Security Results

**Theorem 1.** *The scheme is unforgeable against the type I adversary $\mathcal{A}_1$ if the RSA problem is hard in random oracle model.*

*Proof.* Suppose the challenger $\mathcal{C}$ receives a random instance $(N, b, \mathcal{Y})$ of the RSA problem and has to find an element $\mathcal{X} \in Z_N^*$ such that $\mathcal{X}^b = \mathcal{Y}$. Challenger $\mathcal{C}$ will run $\mathcal{A}_1$ as a subroutine and act as $\mathcal{A}_1$ 's challenger in the UNF-CLMPMS Game I.

**Setup:** At the beginning of the game, $\mathcal{C}$ runs the setup algorithm with the parameter $k$ and gives $\mathcal{A}_1$ the system parameters $params = \{N, b, \mathbb{G}, P, H_0, H_1, H_2, H_3, H_4\}$ and $\mathcal{A}_1$ doesn't know the master secret key $msk = (p, q, a)$.

**Queries:** Without loss of generality we assume that all the queries are distinct and $\mathcal{A}_1$ will make $H_0$ query for $ID_i$ before $ID_i$ is used in any other queries.

1) $H_0$ queries: $\mathcal{C}$ maintains the list $L_0$ of tuple $(ID_i, A_i)$. The list is initially empty. When $\mathcal{A}_1$ makes a query $H_0(ID_i)$, $\mathcal{C}$ responds as follows: At the $j^{th}$ $H_0$ query, $\mathcal{C}$ sets $H_0(ID^*) = \mathcal{Y}$. For $i \neq j$, $\mathcal{C}$ randomly picks a value $A_i \in \mathbb{Z}_N^*$ and sets $H_0(ID_i) = A_i^b$. Then the query and the answer will be stored in the list $L_0$.
2) $H_1$ queries: $\mathcal{C}$ maintains the list $L_1$ of tuple $(\gamma_i, k_i)$. The list is initially empty. When $\mathcal{A}_1$ makes a query $H_1(\gamma_i)$, $\mathcal{C}$ randomly picks a value $k_i \in \mathbb{Z}_b^*$ and sets $H_1(\gamma_i) = k_i$. The query and the answer will be stored in the list $L_1$.
3) $H_2$ queries: $\mathcal{C}$ maintains the list $L_2$ of tuple $(\gamma_i, h_i)$. The list is initially empty. When $\mathcal{A}_1$ makes a query $H_2(\gamma_i)$, $\mathcal{C}$ randomly picks a value $h_i \in \mathbb{Z}_b^*$ and sets $H_2(\gamma_i) = h_i$. The query and the answer will be stored in the list $L_2$.
4) $H_3$ queries: $\mathcal{C}$ maintains the list $L_3$ of tuple $(\eta_i, \alpha_i)$. The list is initially empty. When $\mathcal{A}_1$ makes a query $H_3(\eta_i)$, $\mathcal{C}$ randomly picks a value $\alpha_i \in \mathbb{Z}_b^*$ and sets $H_3(\eta_i) = \alpha_i$. The query and the answer will be stored in the list $L_3$.
5) $H_4$ queries: $\mathcal{C}$ maintains the list $L_4$ of tuple $(\eta_i, \beta_i)$. The list is initially empty. When $\mathcal{A}_1$ makes a query $H_4(\eta_i)$, $\mathcal{C}$ randomly picks a value $\beta_i \in \mathbb{Z}_b^*$ and sets $H_4(\eta_i) = \beta_i$. The query and the answer will be stored in the list $L_4$.
6) User-Public-Key queries: $\mathcal{C}$ maintains the list $L_U$ of tuple $(ID_i, t_i, P_i)$. When $\mathcal{A}_1$ makes user public key query for $ID_i$, $\mathcal{C}$ randomly chooses $t_i \in \mathbb{Z}_b^*$, sets $P_i = t_i P$. Then sends the $P_i$ to $\mathcal{A}_1$. The tuple $(ID_i, t_i, P_i)$ will be stored in the list $L_U$.
7) User-Public-Key-Replacement: $\mathcal{C}$ maintains the list $L_R$ of tuple $(ID_i, P_i, P_i')$. When $\mathcal{A}_1$ makes a user public key replacement for $ID_i$ with a new value $P_i'$, $\mathcal{C}$ replaces the current public key $P_i$ with the value $P_i'$ and the tuple $(ID_i, P_i, P_i')$ will be stored in the list $L_R$.
8) Partial-Private-Key queries: $\mathcal{C}$ maintains the list $L_K$ of tuple $(ID_i, A_i)$. When $\mathcal{A}_1$ makes a partial private key query for $ID_i$, If $ID_i = ID^*$, $\mathcal{C}$ fails and stops, otherwise $\mathcal{C}$ finds the tuple $(ID_i, A_i)$ in list $L_0$ and responds with $A_i$. The tuple $(ID_i, A_i)$ will be stored in the list $L_K$.
9) Secret-Value queries: $\mathcal{C}$ maintains the list $L_S$ of tuple $(ID_i, t_i)$. When $\mathcal{A}_1$ makes a secret value query for $ID_i$, $\mathcal{C}$ finds the tuple $(ID_i, t_i, P_i)$ in list $L_U$ and responds with $t_i$. The tuple $(ID_i, t_i)$ will be stored in the list $L_S$. $\mathcal{A}_1$ can't query the secret value for $ID_i$ whose public key has been replaced.
10) Proxy-Certificate-Generation : When $\mathcal{A}_1$ submits all signers' identities/public keys $(ID_i, P_i)$, $ID_i \in \mathcal{N} \cup \mathcal{L}$ and a warrant $m_\omega$ to the challenger,

$\mathcal{C}$ responds by running the certificate generation algorithm on the warrant $m_\omega$ and the signers' full private key $(t_i, D_i), ID_i \in \mathcal{N} \cup \mathcal{L}$, then outputs a certificate as follows:

If $ID^* \notin \mathcal{N} \cup \mathcal{L}$ and $(\mathcal{N} \cup \mathcal{L}) \bigcap L_R = \emptyset$, $\mathcal{C}$ gives a certificate by calling the certificate generation algorithm. Otherwise, $\mathcal{C}$ does the following:

a. Randomly selects $r \in Z_b^*, R \in Z_N^*$.
b. Randomly selects $k_i, h_i \in Z_b^*$ for each $ID_i \in \mathcal{N} \cup \mathcal{L}$.
c. Computes $S = rP - \sum_{ID_i \in \mathcal{N} \bigcup \mathcal{L}} k_i P_i$ and $T = R^b \cdot \prod_{ID_i \in \mathcal{N} \bigcup \mathcal{L}} Q_i^{-h_i}$.
d. Stores the relation $k_i = H_1(m_\omega, ID_i, P_i, S, T)$ and $h_i = H_2(m_\omega, ID_i, P_i, S, T)$ for each $ID_i \in \mathcal{N} \cup \mathcal{L}$. Repeats the steps (1)-(3) if collision occurs.
e. Outputs $\pi = (m_\omega, r, R, S, T)$ as the proxy certificate.

11) Multi-proxy Multi-sign: When $\mathcal{A}_1$ submits certificate $\pi = (m_\omega, r, R, S, T)$ and a message $M$ to the challenger, $\mathcal{C}$ outputs a signature by running the multi-proxy multi-sign algorithm on $\pi$ and $M$ as follows: If $ID^* \notin \mathcal{L}$ and $\mathcal{L} \bigcap L_R = \emptyset$, $\mathcal{C}$ gives a signature by calling the multi-proxy multi-sign algorithm. Otherwise, $\mathcal{C}$ does the following:

a. Randomly selects $u \in Z_b^*, U \in Z_N^*$.
b. Randomly selects $\alpha_j, \beta_j \in Z_b^*$ for each $ID_j \in \mathcal{L}$.
c. Computes $X = uP - lS - \sum_{i=1}^{n} lk_{oi} P_{oi} - \sum_{j=1}^{l} (lk_{pj} + \alpha_j) P_{pj}$ and $Y = U^b \cdot T^{-l} \cdot \prod_{i=1}^{n} Q_{oi}^{-lh_{oi}} \cdot \prod_{j=1}^{l} Q_{pj}^{-lh_{pj}-\beta_j}$.
d. Stores the relation $\alpha_j = H_3(M, m_\omega, ID_{p_j}, P_{p_j}, S, T, X, Y)$ and $\beta_j = H_4(M, m_\omega, ID_{p_j}, P_{p_j}, S, T, X, Y)$. Repeats the steps (1)-(3) if collision occurs.
e. Outputs $\sigma = (M, m_\omega, u, U, S, T, X, Y)$ as the proxy signature.

**Forge:** $\mathcal{A}_1$ outputs the tuple $\{\pi = (m_\omega, r, R, S, T), \mathcal{L} \cup \mathcal{N}, \bigcup_{ID_i \in \mathcal{L} \cup \mathcal{N}} P_i\}$ or $\{\sigma = (M, m_\omega, u, U, S, T, X, Y), \mathcal{L} \cup \mathcal{N}, \bigcup_{ID_i \in \mathcal{L} \cup \mathcal{N}} P_i\}$.

**Solve RSAP.** If $\mathcal{A}_1$'s output satisfies none of the following 3 cases in UNF-CLMPMS Game I, $\mathcal{C}$ aborts. Otherwise, $\mathcal{C}$ can solve the RSA problem as follows:

**Case 1.** The final output is $\{\pi = (m_\omega, r, R, S, T), \mathcal{L} \cup \mathcal{N}, \bigcup_{ID_i \in \mathcal{L} \cup \mathcal{N}} P_i\}$ and the output satisfies the requirement of Case 1 as defined in UNF-CLMPMS Game I. In fact, $\pi$ is the signature on $m_\omega$. If $ID^* \in \mathcal{N} \cup \mathcal{L}$, we can solve the RSA problem as follows.

Without loss of generality, we may assume that $ID^* = ID_\lambda$. By Forking Lemma for generic signature scheme, we can get another $\pi' = (m_\omega, r, R', S, T)$. To do so we maintain all the random tapes in two invocations are the same except the $\lambda^{th}$ result returned by $H_2$ query of the forged message. In other words $h_\lambda \neq h'_\lambda$ and $h_i = h'_i$ for $i \neq \lambda$. The relation becomes $(R' \cdot R^{-1})^b = \mathcal{Y}^{h'_\lambda - h_\lambda} \bmod N$. Since $h_\lambda, h'_\lambda \in \mathbb{Z}_b^*$, we have that $|h'_\lambda - h_\lambda| < b$. By the element $b$ is a prime number, then $gcd(b, h'_\lambda - h_\lambda) = 1$. This means that there exists two integers $\mu, \nu$ such that $\mu b + \nu(h'_\lambda - h_\lambda) = 1$. Finally, the value $\mathcal{X} = (R' R^{-1})^\nu \mathcal{Y}^\mu \bmod N$ is the solution of the given instance of the RSA problem.

$$\begin{aligned} \mathcal{X}^b &= (R' R^{-1})^{b\nu} \mathcal{Y}^{b\mu} \\ &= \mathcal{Y}^{\nu(h'_\lambda - h_\lambda)} \mathcal{Y}^{b\mu} \\ &= \mathcal{Y}^{b\mu + \nu(h'_\lambda - h_\lambda)} \\ &= \mathcal{Y}. \end{aligned}$$

**Probability of success.** Let $q_{H_i}$ $(i = 0, 1, 2, 3, 4)$, $q_U$, $q_K$, $q_C$ and $q_P$ be the number of $H_i (i = 0, 1, 2, 3, 4)$ queries, user public key queries, partial private key queries, proxy certificate generation queries, multi-proxy multi-signature queries, respectively.

The probability that $\mathcal{C}$ doesn't fail during the queries is $\frac{q_{H_0} - q_K}{q_{H_0}}$. The probability that $ID^* \in \mathcal{L} \cup \mathcal{N}$ is $\frac{n+l-1}{q_K} \cdot \frac{1}{q_{H_0} - q_K}$. So the combined probability is $\frac{q_{H_0} - q_K}{q_{H_0}} \cdot \frac{n+l-1}{q_K} \cdot \frac{1}{q_{H_0} - q_K} = \frac{n+l-1}{q_K \cdot q_{H_0}}$. Therefore, if $\mathcal{A}_1$ can succeed with the probability $\varepsilon$ within time $\mathcal{T}$, then $\mathcal{C}$ can solve RSAP with the probability $\frac{(n+l-1)\varepsilon}{q_K \cdot q_{H_0}}$. The running time required for $\mathcal{C}$ is: $2\mathcal{T} + [q_{H_0} + (3n + 3l + 2)q_D + 2lq_P]T_N + [q_U + (2n + 2l + 2)q_D + lq_P)]T_E$, where $T_N$ denotes the time for a modular operation and $T_E$ denotes the time for a exponentiation in $\mathbb{G}$.

**Case 2.** The final output is $\{\sigma = (M, m_\omega, u, U, S, T, X, Y), \mathcal{L} \cup \mathcal{N}, \bigcup_{ID_i \in \mathcal{L} \cup \mathcal{N}} P_i\}$ and the output satisfies the requirement of Case 2 as defined in UNF-CLMPMS Game I. If $ID^* \in \mathcal{N}$, we can solve the RSA problem as follows.

Without loss of generality, we may assume that $ID^* = ID_\lambda$. By the Forking Lemma for generic signature scheme, we can get another signature $(M, m_\omega, u, U', S, T, X, Y)$. To do so we maintain all the random tapes in two invocations are the same except the $\lambda^{th}$ result returned by $H_2$ query of the forged message. In other words $h_\lambda \neq h'_\lambda$ and $h_i = h'_i$ for $i \neq \lambda$. The relation becomes $(U' \cdot U^{-1})^{bl^{-1}} = \mathcal{Y}^{h'_\lambda - h_\lambda} \bmod N$. Since $h_\lambda, h'_\lambda \in \mathbb{Z}_b^*$, we have that $|h'_\lambda - h_\lambda| < b$. By the element $b$ is a prime number, then $gcd(b, h'_\lambda - h_\lambda) = 1$. This means that there exists two integers $\mu, \nu$ such that $\mu b + \nu(h'_\lambda - h_\lambda) = 1$. Finally, the value $\mathcal{X} = (U' U^{-1})^{l^{-1}\nu} \mathcal{Y}^\mu \bmod N$ is the solution of

the given instance of the RSA problem.

$$\begin{aligned}\mathcal{X}^b &= (U'U^{-1})^{bl^{-1}\nu}\mathcal{Y}^{b\mu}\\ &= \mathcal{Y}^{\nu(h'_\lambda - h_\lambda)}\mathcal{Y}^{b\mu}\\ &= \mathcal{Y}^{b\mu+\nu(h'_\lambda-h_\lambda)}\\ &= \mathcal{Y}.\end{aligned}$$

**Probability of success.** Let $q_{H_i}$ $(i = 0, 1, 2, 3, 4)$, $q_U$, $q_K$, $q_C$ and $q_P$ be the number of $H_i(i = 0, 1, 2, 3, 4)$ queries, user public key queries, partial private key queries, proxy certificate generation queries, multi-proxy multi-signature queries, respectively.

The probability that $\mathcal{C}$ doesn't fail during the queries is $\frac{q_{H_0}-q_K}{q_{H_0}}$. The probability that $ID^* \in \mathcal{N}$ is $\frac{n-1}{q_K}\cdot\frac{1}{q_{H_0}-q_K}$. So the combined probability is $\frac{q_{H_0}-q_K}{q_{H_0}}\cdot\frac{n-1}{q_K}\cdot\frac{1}{q_{H_0}-q_K} = \frac{n-1}{q_K\cdot q_{H_0}}$. Therefore, if $\mathcal{A}_1$ can succeed with the probability $\varepsilon$ within time $\mathcal{T}$, then $\mathcal{C}$ can solve RSAP with the probability $\frac{(n-1)\varepsilon}{q_K\cdot q_{H_0}}$. The running time required for $\mathcal{C}$ is the same as the time in Case 1.

**Case 3.** The final output is $\{\sigma = (M, m_\omega, u, U, S, T, X, Y), \mathcal{L}\cup\mathcal{N}, \bigcup_{ID_i\in\mathcal{L}\cup\mathcal{N}} P_i\}$ and the output satisfies the requirement of *Case 3* as defined in UNF-CLMPMS Game I. If $ID^* \in \mathcal{L}$, we can solve the RSA problem as follows.

Without loss of generality, we may assume that $ID^* = ID_\lambda$. By the Forking Lemma for generic signature scheme, we can get another signature $(M, m_\omega, u, U', S, T, X, Y)$. To do so we maintain all the random tapes in two invocations are the same except the $\lambda^{th}$ result returned by $H_4$ query of the forged message. In other words $\beta_\lambda \neq \beta'_\lambda$ and $\beta_j = \beta'_j$ for $j \neq \lambda$. The relation becomes $(U'\cdot U^{-1})^b = \mathcal{Y}^{\beta'_\lambda-\beta_\lambda} \bmod N$. Since $\beta_\lambda, \beta'_\lambda \in \mathbb{Z}^*_b$, we have that $|\beta'_\lambda - \beta_\lambda| < b$. By the element $b$ is a prime number, then $gcd(b, \beta'_\lambda - \beta_\lambda) = 1$. This means that there exists two integers $\mu, \nu$ such that $\mu b + \nu(\beta'_\lambda - \beta_\lambda) = 1$. Finally, the value $\mathcal{X} = (U'U^{-1})^\nu\mathcal{Y}^\mu \bmod N$ is the solution of the given instance of the RSA problem.

$$\begin{aligned}\mathcal{X}^b &= (U'U^{-1})^{b\nu}\mathcal{Y}^{b\mu}\\ &= \mathcal{Y}^{\nu(\beta'_\lambda-\beta_\lambda)}\mathcal{Y}^{b\mu}\\ &= \mathcal{Y}^{b\mu+\nu(\beta'_\lambda-\beta_\lambda)}\\ &= \mathcal{Y}.\end{aligned}$$

**Probability of success.** Let $q_{H_i}(i = 0, 1, 2, 3, 4)$, $q_U$, $q_K$, $q_C$ and $q_P$ be the number of $H_i(i = 0, 1, 2, 3, 4)$ queries, user public key queries, partial private key queries, proxy certificate generation queries, multi-proxy multi-signature queries, respectively.

The probability that $\mathcal{C}$ doesn't fail during the queries is $\frac{q_{H_0}-q_K}{q_{H_0}}$. The probability that $ID^* \in \mathcal{L}$ is $\frac{l-1}{q_K}\cdot\frac{1}{q_{H_0}-q_K}$. So the combined probability is $\frac{q_{H_0}-q_K}{q_{H_0}}\cdot\frac{l-1}{q_K}\cdot\frac{1}{q_{H_0}-q_K} = \frac{l-1}{q_K\cdot q_{H_0}}$. Therefore, if $\mathcal{A}_1$ can succeed with the probability $\varepsilon$ within time $\mathcal{T}$, then $\mathcal{C}$ can solve RSAP with the probability $\frac{(l-1)\varepsilon}{q_K\cdot q_{H_0}}$. The running time required for $\mathcal{C}$ is the same as the time in Case 1.

□

**Theorem 2.** *The scheme is unforgeable against the type II adversary $\mathcal{A}_2$ if the DL problem is hard in randomly oracle model.*

*Proof.* Suppose the challenger $\mathcal{C}$ receives a random instance $(P, xP)$ of the DL problem and has to compute $x \in Z^*_b$. Challenger $\mathcal{C}$ will run $\mathcal{A}_2$ as a subroutine and act as $\mathcal{A}_2$ 's challenger in the UNF-CLMPMS Game II.

**Setup:** At the beginning of the game, $\mathcal{C}$ runs the setup algorithm with the parameter $k$ and gives $\mathcal{A}_2$ the system parameters $params = \{N, b, \mathbb{G}, P, H_0, H_1, H_2, H_3, H_4\}$ and the master secret key $msk = (p, q, a)$.

**Queries:** Without loss of generality we assume that all the queries are distinct and $\mathcal{A}_2$ will make user's public key query for $ID_i$ before $ID_i$ is used in any other queries.

1) User-Public-Key queries: $\mathcal{C}$ maintains the list $L_U$ of tuple $(ID_i, t_i, P_i)$. When $\mathcal{A}_2$ makes public key query for $ID_i$, $\mathcal{C}$ responds as follows:

   At the $j^{th}$ query, $\mathcal{C}$ sets $ID_j = ID^*$ and $P^* = xP$. For $i \neq j$, $\mathcal{C}$ randomly chooses $t_i \in \mathbb{Z}^*_b$, sets $P_i = t_iP$. Then the query and the answer will be stored in the list $L_U$.

2) $H_0$ queries: $\mathcal{C}$ maintains the list $L_0$ of tuple $(ID_i, A_i)$. The list is initially empty. When $\mathcal{A}_2$ makes a query $H_0(ID_i)$, $\mathcal{C}$ randomly picks a value $A_i \in \mathbb{Z}^*_N$ and sets $H_0(ID_i) = A^b_i$. Then the query and the answer will be stored in the list $L_0$.

3) $H_1, H_2, H_3, H_4$ queries and User-Public-Key-Replacement are the same as those in Theorem 1.

4) Partial-Private-Key queries: $\mathcal{C}$ maintains the list $L_K$ of tuple $(ID_i, A_i)$. When $\mathcal{A}_2$ makes a partial private key query for $ID_i$, $\mathcal{C}$ finds the tuple $(ID_i, A_i)$ in list $L_0$ and responds with $A_i$. The tuple $(ID_i, A_i)$ will be stored in the list $L_K$.

5) Secret-Value queries: $\mathcal{C}$ maintains the list $L_S$ of tuple $(ID_i, t_i)$. When $\mathcal{A}_2$ makes a secret value query for $ID_i$, If $ID_i = ID^*$, $\mathcal{C}$ fails and stops, otherwise $\mathcal{C}$ finds the tuple $(ID_i, t_i, P_i)$ in list $L_U$ and responds with $t_i$. The tuple $(ID_i, t_i)$ will be stored in the list $L_S$. $\mathcal{A}_2$ can't query the secret value for $ID_i$ whose public key has been replaced.

6) proxy certificate generation queries and Multi-proxy Multi-sign queries are the same as those in Theorem 1.

**Forge:** $\mathcal{A}_2$ outputs the tuple $\{\pi = (m_\omega, r, R, S, T), \mathcal{L} \cup \mathcal{N}, \bigcup_{ID_i \in \mathcal{L} \cup \mathcal{N}} P_i\}$ or $\{\sigma = (M, m_\omega, u, U, S, T, X, Y), \mathcal{L} \cup \mathcal{N}, \bigcup_{ID_i \in \mathcal{L} \cup \mathcal{N}} P_i\}$.

**Solve DLP.** If $\mathcal{A}_2$'s output satisfies none of the following 3 cases in UNF-CLMPMS Game II, $\mathcal{C}$ aborts. Otherwise, $\mathcal{C}$ can solve the DL problem as follows:

**Case 1.** The final output is $\{\pi = (m_\omega, r, R, S, T), \mathcal{L} \cup \mathcal{N}, \bigcup_{ID_i \in \mathcal{L} \cup \mathcal{N}} P_i\}$ and the output satisfies the requirement of Case 1 as defined in UNF-CLMPMS Game II. In fact, $\pi$ is the signature on $m_\omega$. If $ID^* \in \mathcal{N} \cup \mathcal{L}$, we can solve the DL problem as follows.

Without loss of generality, we may assume that $ID^* = ID_\lambda$. By Forking Lemma for generic signature scheme, we can get another $\pi'(m_\omega, r', R, S, T)$. To do so we maintain all the random tapes in two invocations are the same except the $\lambda^{th}$ result returned by $H_1$ query of the forged message. In other words $k_\lambda \neq k'_\lambda$ and $k_i = k'_i$ for $i \neq \lambda$. We note that $r = c_\lambda + k_\lambda x + \sum_{ID_i \in \mathcal{N} \cup \mathcal{L} \setminus \{ID*\}} (c_i + k_i t_i)$, $r' = c_\lambda + k'_\lambda x + \sum_{ID_i \in \mathcal{N} \cup \mathcal{L} \setminus \{ID*\}} (c_i + k_i t_i)$. It follows that $x = \frac{r - r'}{k_\lambda - k'_\lambda}$.

**Probability of success.** Let $q_{H_i} (i = 0, 1, 2, 3, 4)$, $q_U$, $q_K$, $q_S$, $q_C$ and $q_P$ be the number of $H_i (i = 0, 1, 2, 3, 4)$ queries, user public key queries, partial private key queries, secret value queries, proxy certificate generation queries, multi-proxy multi-signature queries, respectively.

The probability that $\mathcal{C}$ doesn't fail during the queries is $\frac{q_U - q_S}{q_U}$. The probability that $ID^* \in \mathcal{L} \cup \mathcal{N}$ is $\frac{n+l-1}{q_S} \cdot \frac{1}{q_U - q_S}$. So the combined probability is $\frac{q_U - q_S}{q_U} \cdot \frac{n+l-1}{q_S} \cdot \frac{1}{q_U - q_S} = \frac{n+l-1}{q_S \cdot q_U}$. Therefore, if $\mathcal{A}_2$ can succeed with the probability $\varepsilon$ within time $\mathcal{T}$, then $\mathcal{C}$ can solve DL problem with the probability $\frac{(n+l-1)\varepsilon}{q_S \cdot q_U}$. The running time required for $\mathcal{C}$ is: $2\mathcal{T} + [q_{H_0} + (3n + 3l + 2)q_D + 2lq_P]T_N + [q_U + (2n + 2l + 2)q_D + lq_P)]T_E$, where $T_N$ denotes the time for a modular operation and $T_E$ denotes the time for a exponentiation in $\mathbb{G}$.

**Case 2.** The final output is $\{\sigma = (M, m_\omega, u, U, S, T, X, Y), \mathcal{L} \cup \mathcal{N}, \bigcup_{ID_i \in \mathcal{L} \cup \mathcal{N}} P_i\}$ and the output satisfies the requirement of Case 2 as defined in UNF-CLMPMS Game II. If $ID^* \in \mathcal{N}$, we can solve the DL problem as follows.

Without loss of generality, we may assume that $ID^* = ID_\lambda$. By the Forking Lemma for generic signature scheme, we can get another signature $(M, m_\omega, u', U, S, T, X, Y)$. To do so we maintain all the random tapes in two invocations are the same except the $\lambda^{th}$ result returned by $H_1$ query of the forged message. In other words $k_\lambda \neq k'_\lambda$ and $k_i = k'_i$ for $i \neq \lambda$. We note that $u = l(c_\lambda + k_\lambda x + \sum_{ID_i \in \mathcal{N} \cup \mathcal{L} \setminus \{ID*\}} (c_i + k_i t_i)) + \sum_{ID_j \in \mathcal{L}} (a_j + \alpha_j t_j)$, $u' = l(c_\lambda + k'_\lambda x + \sum_{ID_i \in \mathcal{N} \cup \mathcal{L} \setminus \{ID*\}} (c_i + k_i t_i)) + \sum_{ID_j \in \mathcal{L}} (a_j + \alpha_j t_j)$. It follows that $x = \frac{u - u'}{l(k_\lambda - k'_\lambda)}$.

**Probability of success.** Let $q_{H_i} (i = 0, 1, 2, 3, 4)$, $q_U$, $q_K$, $q_S$, $q_C$ and $q_P$ be the number of $H_i (i = 0, 1, 2, 3, 4)$ queries, user public key queries, partial private key queries, secret value queries, proxy certificate generation queries, multi-proxy multi-signature queries, respectively.

The probability that $\mathcal{C}$ doesn't fail during the queries is $\frac{q_U - q_S}{q_U}$. The probability that $ID^* \in \mathcal{N}$ is $\frac{n-1}{q_S} \cdot \frac{1}{q_U - q_S}$. So the combined probability is $\frac{q_U - q_S}{q_U} \cdot \frac{n-1}{q_S} \cdot \frac{1}{q_U - q_S} = \frac{n-1}{q_S \cdot q_U}$. Therefore, if $\mathcal{A}_2$ can succeed with the probability $\varepsilon$ within time $\mathcal{T}$, then $\mathcal{C}$ can solve DL problem with the probability $\frac{(n-1)\varepsilon}{q_S \cdot q_U}$. The running time required for $\mathcal{C}$ is the same as the time in Case 1.

**Case 3.** The final output is $\{\sigma = (M, m_\omega, u, U, S, T, X, Y), \mathcal{L} \cup \mathcal{N}, \bigcup_{ID_i \in \mathcal{L} \cup \mathcal{N}} P_i\}$ and the output satisfies the requirement of *Case 3* as defined in UNF-CLMPMS Game II. If $ID^* \in \mathcal{L}$, we can solve the DL problem as follows.

Without loss of generality, we may assume that $ID^* = ID_\lambda$. By the Forking Lemma for generic signature scheme, we can get another signature $(M, m_\omega, u', U, S, T, X, Y)$. To do so we maintain all the random tapes in two invocations are the same except the $\lambda^{th}$ result returned by $H_3$ query of the forged message. In other words $\alpha_\lambda \neq \alpha'_\lambda$ and $\alpha_j = \alpha'_j$ for $j \neq \lambda$. We note that $u = lr + \sum_{ID_j \in \mathcal{L} \setminus \{ID^*\}} (a_j + \alpha_j t_j) + (a_\lambda + \alpha_\lambda x)$, $u' = lr + \sum_{ID_j \in \mathcal{L} \setminus \{ID^*\}} (a_j + \alpha_j t_j) + (a_\lambda + \alpha'_\lambda x)$, It follows that $x = \frac{u - u'}{\alpha_\lambda - \alpha'_\lambda}$.

**Probability of success.** Let $q_{H_i} (i = 0, 1, 2, 3, 4)$, $q_U$, $q_K$, $q_S$, $q_C$ and $q_P$ be the number of $H_i (i = 0, 1, 2, 3, 4)$ queries, user public key queries, partial private key queries, secret value queries, proxy certificate generation queries, multi-proxy multi-signature queries, respectively.

The probability that $\mathcal{C}$ doesn't fail during the queries is $\frac{q_U - q_S}{q_U}$. The probability that $ID^* \in \mathcal{L}$ is $\frac{l-1}{q_S} \cdot \frac{1}{q_U - q_S}$. So the combined probability is $\frac{q_U - q_S}{q_U} \cdot \frac{l-1}{q_S} \cdot \frac{1}{q_U - q_S} = \frac{l-1}{q_S \cdot q_U}$. Therefore, if $\mathcal{A}_2$ can succeed with the probability $\varepsilon$ within time $\mathcal{T}$, then $\mathcal{C}$ can solve DL Problem with the probability $\frac{(l-1)\varepsilon}{q_S \cdot q_U}$. The running time required for $\mathcal{C}$ is the same as the time in Case 1.

□

## 5 Efficiency and Comparison

Our scheme is constructed without using bilinear pairing. In the following, we compare the performance of our scheme with several MPMS schemes in Table 2. We define some notations as follows:

- $T_P$: A pairing operation.
- $E_P$: A pairing-based scalar multiplication operation.
- $T_E$: A scalar multiplication operation in the elliptic curve group $\mathbb{G}$.
- $T_N$: A modular exponent operation in $\mathbb{Z}_N$.

Through PIV 3-GHZ processor with 512-MB memory and a Windows XP operation system. He *et al.* [5] obtained the running time for cryptographic operations. To achieve 1024-bit RSA level security, they use the Tate pairing defined over a super singular curve $E/F_p : y^2 = x^3 + x$ with embedding degree 2, where $q$ is a 160-bit Solinas prime $q = 2^{159} + 2^{17} + 1$ and $p$ is a 512-bit prime satisfying $p + 1 = 12qr$. To achieve the same security level, they employed the parameter secp160r1 [18], where $p = 2^{160} - 2^{31} - 1$. The running times are listed in Table 1.

Table 1: Cryptographic operation time (in milliseconds)

| $T_P$ | $T_N$ | $E_P$ | $T_E$ |
|---|---|---|---|
| 20.04 | 5.31 | 6.38 | 2.21 |

To evaluate the computation efficiency of different schemes, we use a simple method. For example in [10], system costs $3n+3l+2$ pairing-based scalar multiplication operations and $3n+3l$ pairing operations in Proxy Certificate Generation, system costs $3l + 3$ pairing-based scalar multiplication operations and $3l + 6$ pairing operations in Multi-Proxy Multi-Sign and Verify. Hence system costs $3n + 6l + 5$ pairing-based scalar multiplication operations and $3n+6l+6$ pairing operations in total. To facilitate the comparison, we let $n = l = 10$. So the resulting computation time is $95 \times 6.38 + 96 \times 11.20 = 2721.34$. The detailed comparison results of several different MPMS schemes are illustrated in Table 2 and Table 3.

Table 2: Comparison of several CLMPMS schemes

| Scheme | Public key form | Secure base |
|---|---|---|
| Li [10] | ID-base | CDHP |
| Sahu [15] | ID-base | CDHP |
| Our scheme | Certificateless | RSAP and DLP |

## 6 Conclusion

In a multi-proxy multi-signature scheme, the group of original signers delegate their signing rights to the proxy group. RSA is a key cryptography technique and provides various interfaces for the applied software in real-life scenarios. Although some good results were achieved in speeding up the computation of pairing function in recent years, the computation cost of the pairing is much higher than that of the exponentiation in a RSA group and also much higher than the scalar multiplication over the elliptic curve group. In this paper, we propose a certificateless multi-proxy multi-signature scheme and prove that our scheme is unforgeable under the strongest security model where the Type I/II adversary is a super Type I/II adversary. The analysis shows that our scheme is more efficient than the related schemes. Due to the very good properties of our scheme, it is very useful for practical applications.

## Acknowledgments

This research is supported by the National Natural Science Foundation of China under Grants 61562012, 11261060, the Innovation Group Major Research Projects of Department of Education of Guizhou Province under Grant No.KY[2016]026. The authors gratefully acknowledge the anonymous reviewers for their valuable comments.

## References

[1] S. Al-Riyami and K. Paterson, "Certificateless public key cryptography," in *Advances in Cryptology (Asiacrypt'03)*, LNCS 2894, pp. 452-473, 2003.

[2] M. K. Chande, C. C. Lee, C. T. Li, "Message recovery via an efficient multi-proxy signature with self-certified keys," *International Journal of Network Security*, vol. 19, no. 3, pp. 340-346, 2017.

[3] L. Deng, H. Huang and Y. Qu, "Identity based proxy signature from RSA without pairings," *International Journal of Network Security*, vol. 19, no. 2, pp.229-235, 2017.

[4] L. Deng, J. Zeng and Y. Qu, "Certificateless proxy signature from RSA," *Mathematical Provlems in Engineering*, vol. 2014, pp. 1-10, 2014.

[5] D. He, J. Chen and J. Hu, "An ID-based proxy signature schemes without bilinear pairings," *Annals of Telecommunications*, vol. 66, pp. 657-662, 2011.

[6] D. He, Y. Chen and J. Chen, "An efficient certificateless proxy signature scheme without pairing," *Mathmatical and Computer Modelling*, vol. 57, pp. 2510-2518, 2013.

[7] J. Herranz, "Identity-based ring signatures from RSA," *Theoretical Computer Science*, vol. 389, pp. 100-117, 2007.

Table 3: Comparison of several CLMPMS schemes

| Scheme | Proxy certificate generation | MPMSign and verify | Time(n=l=10) |
|---|---|---|---|
| Li [10] | $(3n+3l+2)E_P+(3n+3l)T_P$ | $(3l+3)E_P+(3l+6)T_P$ | 2721.34 |
| Sahu [15] | $(2n+2l+2)E_P+(3n+3l)T_P$ | $(2l+3)E_P+(3l+6)T_P$ | 2338.54 |
| Our scheme | $(3n+3l)T_E+(4n+4l+4)T_N$ | $(2n+7l+2)T_E+(2n+7l+5)T_N$ | 1275.19 |


[8] K. Kim, D. Hong, and I. Jeong, "Identity-based proxy signature from lattices," *Journal of Communications and Networks*, vol. 15, no. 1, pp. 1-7, 2013.

[9] L. H. Li, S. F. Tzeng, M. S. Hwang, "Generalization of proxy signature based on discrete logarithms", *Computers & Security*, vol. 22, no. 3, pp. 245–255, 2003.

[10] X. Li and K, Chen, "ID-based multi-proxy signature, proxy multi-signature and multi-proxy multi-signature schemes from bilinear pairings," *Applied Mathematics and Computation*, vol. 169, pp. 437-450, 2005.

[11] E. J. L. Lu, M. S. Hwang, and C. J. Huang, "A new proxy signature scheme with revocation", *Applied Mathematics and Computation*, vol. 161, no. 3, PP. 799-806, Feb. 2005.

[12] Y. Lu and J. Li, "Provably secure certificateless proxy signature scheme in the standard model," *Theoretical Computer Science*, vol. 639, pp. 42-59, 2016.

[13] M. Mambo, K. Usuda, and E. Okamoto, "Proxy signatures: Delegation of the power to sign messages," *IEICE Transactions of The Fundamentals of Electronics, Communications and Computer Sciences*, vol. 79, no. 9, pp. 1338-1353, 1996.

[14] S. Mashhadi, "A novel non-repudiable threshold proxy signature scheme with known signers," *International Journal of Network Security*, vol. 15, no. 4, pp. 274-279, 2013.

[15] R. Sahu and S. Padhye, "An ID-based multi-proxy multi-signature scheme," *International Conference on Computer and Communication Technology*, pp. 60-63, 2010.

[16] A. Shamir, "Identity-based cryptosystem and signature scheme," in *Advances in Cryptology (Crypto'84)*, LNCS, vol. 196, pp. 47-53, 1984.

[17] N. Tiwari and S. Padhye, "Provable secure multi-proxy signature scheme without bilinear maps," *International Journal of Network Security*, vol. 17, no. 6, pp. 736-742, 2015.

[18] The Certicom Corporation, *SEC2: Recommended Elliptic Curve Domain Parameters*, Dec. 23, 2017. (`www.secg.org/collateral/sec2-final.pdf`)

[19] S. F. Tzeng, M. S. Hwang, C. Y. Yang, "An improvement of nonrepudiable threshold proxy signature scheme with known signers", *Computers & Security*, vol. 23, no. 2, pp. 174-178, Apr. 2004.

[20] S. F. Tzeng, C. Y. Yang, M. S. Hwang, "A Nonrepudiable Threshold Multi-Proxy Multi-Signature Scheme with Shared Verification", *Future Generation Computer Systems*, vol. 20, no. 5, PP. 887-893, June 2004.

[21] J. Xun H. SUN, Q. Wen and H. Zhang, "Improved certificateless multi-proxy signature," *The Journal of China Universities of Posts and Telecommunications*, vol. 19, no. 4, pp. 94-105, 2012.

[22] H. Xiong, F. Li and Z. Qin, "A provably secure proxy signature scheme in certificateless cryptography," *Informatica*, vol. 21, no. 2, pp. 277-294, 2010.

[23] L. Zhang, F. Zhang and Q. Wu, "Delegation of signing rights using certificateless proxy signatures," *Information Sciences*, vol. 184, pp. 298-309, 2012.


## Biography

**Rena Ehmet** received her B.S. from Xinjiang University, Urumqi, China, in 2005; M.S. from Xinjiang University, Urumqi, China, in 2008. She is now a Ph.D candidate in the School of Mathematical Sciences, Xiamen University, Xiamen, China, and she is a teacher in the School of Mathematical Sciences, Xinjiang Normal University, Xinjiang, China. Her recent research interests include cryptography and information safety.

**Lunzhi Deng** received his B.S. from Guizhou Normal University, Guiyang, China, in 2002; M.S. from Guizhou Normal University, Guiyang, China, in 2008; and Ph.D. from Xiamen University, Xiamen, China, in 2012. He is currently a professor in the School of Mathematical Sciences, Guizhou Normal University, Guiyang, China. His recent research interests include cryptography and information security.

**Yingying Zhang** received her B.S. from Xinjiang Normal University, Urumqi, PR China, in 2011; M.S. from Xinjiang Normal University, Urumqi, China, in 2014. She is now a Ph.D candidate in the School of Mathematical Sciences, Xiamen University, Xiamen, China. Her recent research interests include cryptography and information safety.

**Jiwen Zeng** received his B.S. from Gannan Normal University, Ganzhou, China, in 1981; M.S. from Wuhan University, Wuhan, China, in 1988; and Ph.D from Peking University, Beijing, China, in 1995. He is currently a professor in the School of Mathematical Sciences, Xiamen University, Xiamen, China. His recent research interests include group and cryptography.